The principles used in high speed techniques of serial data transmissions

The principles used in SATA interface

Lecture goals:

The explanation of basic SATA principles

The explanation of 8b/10b encoding principles

Making SATA resistant against errors.

The schedule of the lecture:

- Parallel v serial transmissions, former situation (25 years ago) v. the situation today.
- How to synchronize data transmitted serially.
- How to remove possible errors in serial data transmission.
- Characters encoding.
- The importance of zero level of the transmitted signal.

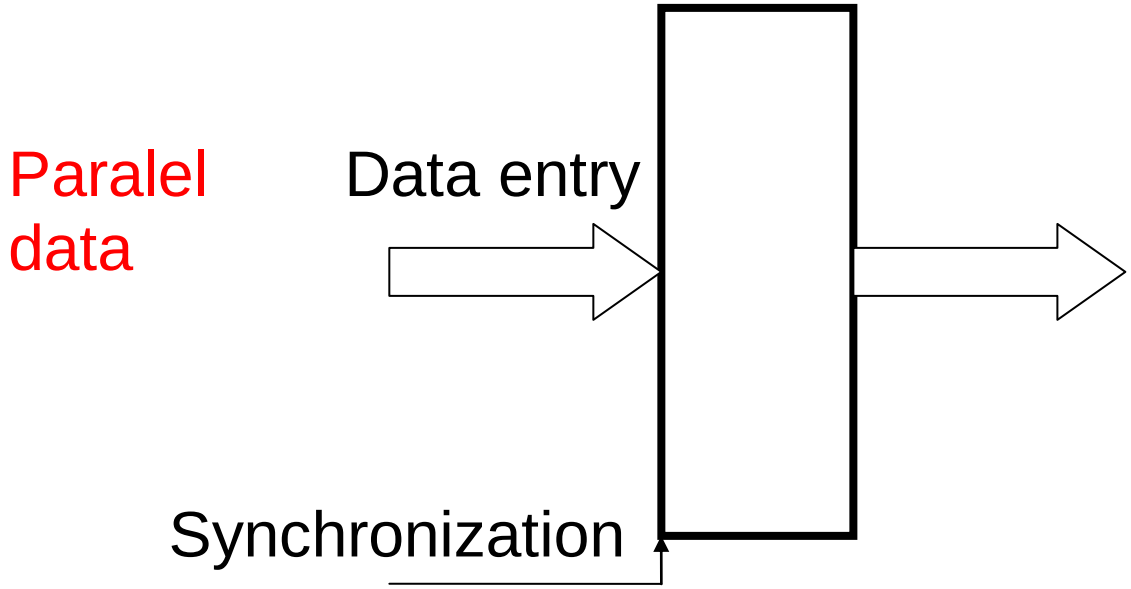
SATA – serial ATA

ATA – AT attachment (parallel interface of disks in PC architecture)

- **Paralel v. serial data transmissions**

  - The situation 25 years ago: parallel principles of data transmissions were faster from the data transmission speed point of view (the volume of data / s) – just because they were parallel (more bits transmitted in parallel).
  - The situation today: the serial principles are seen as those with perspective – it is sure that parameters of these techniques will be further improved (speed, the amount of data being transmitted, reliability). This will not be the situation with parallel techniques.
  - The parallel techniques are still used but they are on their top.

The possibilities of synchronisation of data transmitted in paralel.



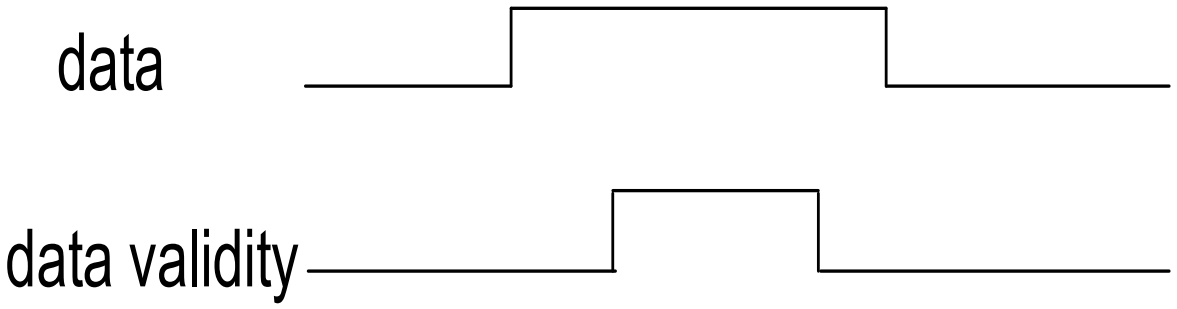
Obr. 1

In the receiver, data must be synchronized. When the frequency increases, then a delay can appear on the data signals against synchronization (clock skew) => an error in data can appear.

The explanation of „clock skew“

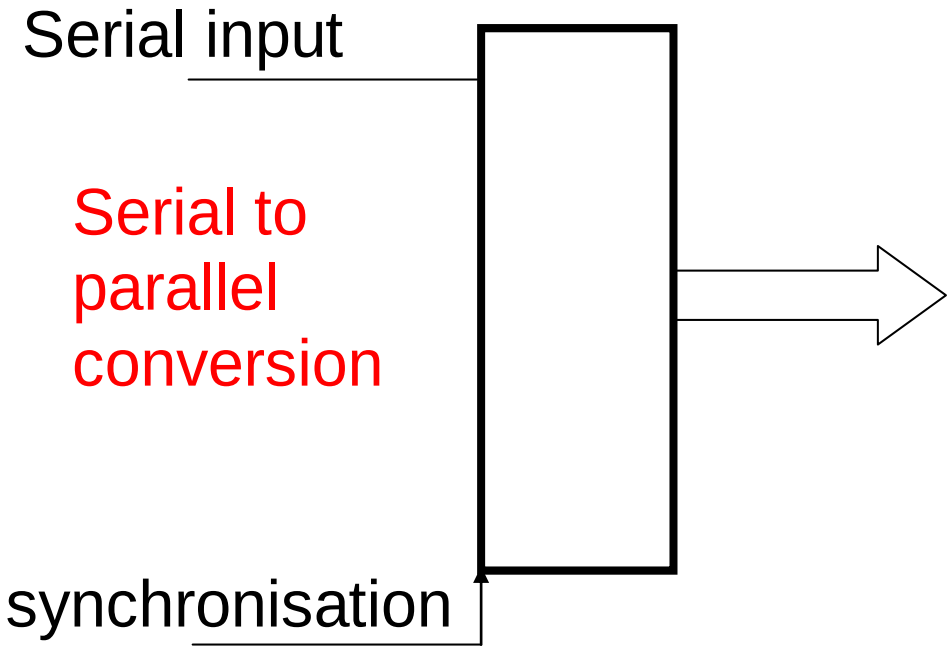
When data are transmitted in parallel (e.g. 32 bits), then when the frequency is increasing the probability increases that one or more bits will not be properly sampled (i.e. loaded to register).

Increasing the frequency / increasing the width of data transmitted in parallel – a problem (e.g. parasite capacitances between data bits).



The synchronization of serially transmitted data

Data in the receiver must be synchronized and converted to parallel form. When the frequency is increased, again the possibility of „clock skew“ occurs.



Obr. 2

Clock skew – the shift of data bits against synchronization.

How to treat „clock skew“

The synchronization will not be transmitted as a lonely signal – it will be derived from data signal or we shall use the internal synchronization (it will be derived from the frequency of data received) – phased-locked loop will be used for this purpose).

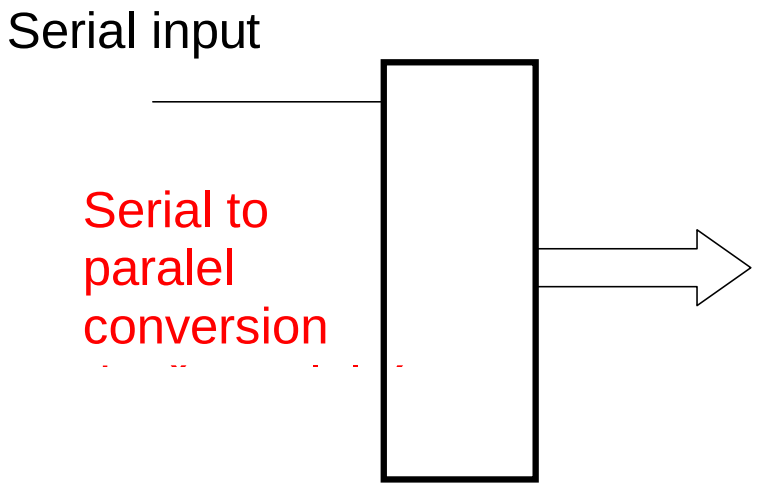


Fig. 3

- The synchronization of accepted data – how to solve it:
  - Just data are in the connection, not the synchronization (the synchronization is not derived from the signal accepted) the receiver generates clock signal of the predefined frequency (it can be done on low frequencies)
  - Synchronization signal belongs to the connection, then the following alternatives exit:

    alternative 1: the synchronization is derived from data,

    alternative 2: for the synchronization an independent connection exists

- **Alternative 1 –** the synchronization is derived from data:
  - If no changes in the data exist, then it is difficult to gain the synchronization from data.

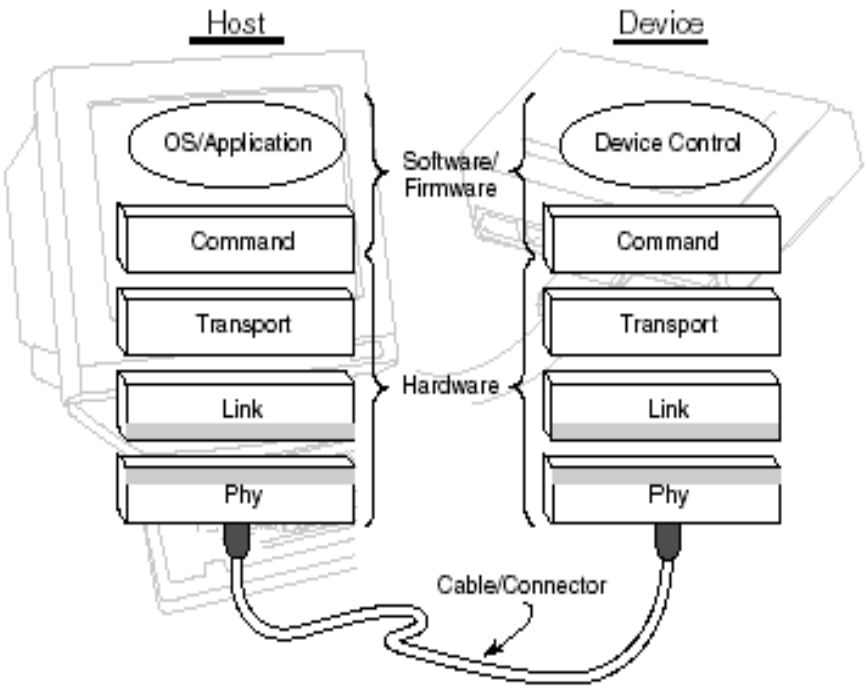
- The feature when the synchronization can be derived by the receiver from data received with the use of phase – locked loop – it is so in SATA and PCI Express.
- The situation when the signal read from the disk has such features that the synchronization can be gained from it then we talk about embedded clock.

- **Alternative 2 –** a separate connection for the synchronization**:**
  - Another connection in the cable and another position in the connector, for the clock signal.
  - Synchronization signal will have a higher frequency than the data transmitted (the synchronization signal will be changing in every bit interval),
  - If data had the following sequence - 010101…, then it will be still the half frequency of the synchronization because it will be changing inside the bit interval,

- The distribution of the synchronization separately from data causes problems in the situation when we need to increase the frequency.
- The problem to sample data correctly when data and clock are separate signals. It might happen that data are not sampled correctly. This is the situation in DVI interface (serial connection where data and synchronization are separate signals).

Conclusion:

- High transmission frequencies – it is not recommended to transfer data and synchronization (clock) along separate signals.
- This condition is satisfied in SATA. SATA interface is not synchronized by pulses transferred as a separate signal.



Obr. 4

- Link layer – responsibility for the encoding of transmitted data, decoding data accepted and for the protocol (see fig. 4).
- Transport layer – developing the format of transmitted data.

- General information – if something is encoded into data in the particular layer, then this type of information is recognised in the receiver on the same layer.

An example of data sample (a high number of changes of the signal being recorded):

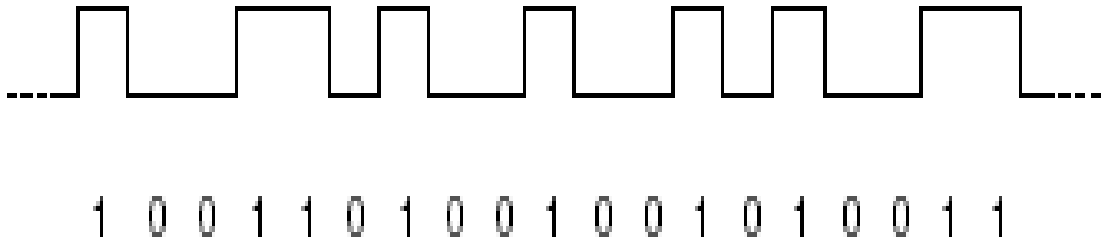


Fig. 5 An illustration example: data with a great number of signal changes

- Explanantion of Fig. 5:

  Bit interval is represented by the width of the shortest interval between changes, then the values in the remaining bits can be recognised.

  Convenient – if the signal representing the data contains so many changes that in the receiver the synchronization can be derived – PLL (Phase Locked Loop) can be used to derive clock. RL (Run

Length) parameter is satisfied. Explanation of RL parameter: the interval in bits for which the signal of data does not change.

An example of data with a low number of chages:

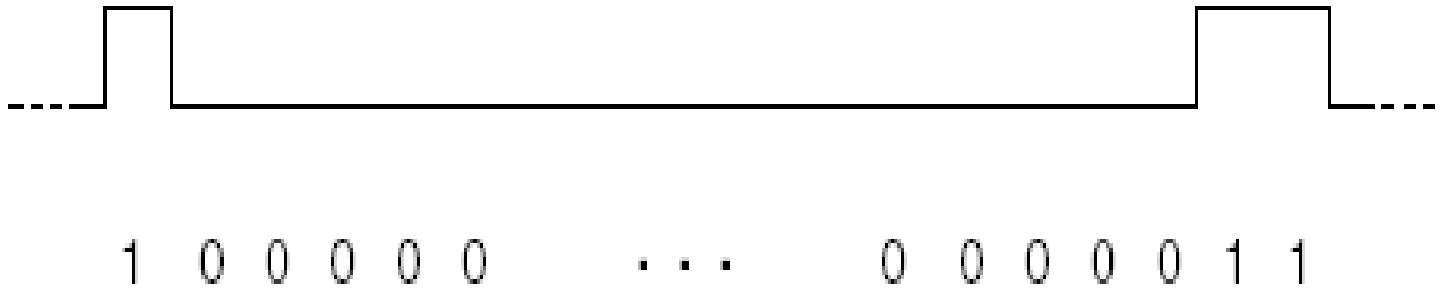


Fig. 6 An illustration example – data with a low number of changes

This data sequence consists of few changes which causes problems during reading – a long interval of „0“

- Comments to Fig. 6:

  As the data accepted contain a low number of signal changes, it will be difficult to derive clock and data compared to signal with a

high number of changes. It is also difficult (if the signal has no changes) to derive how many bits it contains.

- The conclusion:
  - If clock pulses are not encoded in the signal, then changes of signal must exist in the data accepted.
  - Parameter, which represents the number of bits for which the signal is not changing is called Run Length. It must have a predefined value.

- The reflection of these principles in the techniques of serial high speed transmissions (e.g. SATA):
  - What must be achieved: the required value of RL and zero voltage value of the signal transmitted between „0“ and „1“ – the values of voltage transmitted: +- 250 mV).
  - Both requirements are satisfied by 8b/10b encoding.

- The goal of SATA encoding: **1) RL parameter and 2) the same number of „0“ a „1“ in the signal transmitted (zero DC component to be gained)**. DC – Direct Current

- <u>The basic principles of 8b/10b encoding in SATA:</u>

  256 eight bit characters are encoded into 10 bits.

  8 bits offer to create 256 combinations, 10 bits 1024 combinations. From among this set, it is possible to select the number of combinations to satisfy concrete requirements.
- One of the goals of 8b/10b encoding is to have RL patameter on an acceptable level.
- It can be achieved by means of selecting such 10 bit values which satisfy the condition.
- It means that the all „1“ and all „0“ combinations are not utilised. It is possible to select from among 1024 combinations only those 256 values which satisfy the condition. E. g. all „1“ and all „0“ are not used.

- Basic features of SATA:

  - Although the speed of communication is 1,5 Gb/s (now even 3 Gb/s – SATA II and 6 Gb/s – SATA III), the speed modified according to data communication is 150 MB/s (1500/10) and not 187,5 MB/s (1500/8).

- It is caused by encoding principles used in SATA..
  - It is based on 10 bits, not on 8.
  - It is a principle used with high-speed serial communication
  - A disadvantage – the amount of data is reduced by 20%, on the other hand the principle has other advantages (IBM patent 4 486 739 – www.uspto.gov). It is referred to as 8b/10b encoding.

- Another requirement on 8b/10b encoding:
  - **Zero DC (Direct Current) component.**
  - This requirement will be satisfied by equal number of „0“ and „1“ in data.
  - Note: another possibility how to satisfy this requirement is by means of capacitor on the input which blocks DC voltage.
  - A better solution: the same number of „0“ and „1“ in the encoded signal (in SATA it is done in this way).

- Encoding principles – other principles:
  - 10 bits – we have 1024 possible combinations.
  - If we select from among 1024 combinations such combinations which contain the same number of „0“ and „1“ (i.e. five „0“ and five „1“), then we gain just 252 combinations (they have zero DC voltage component).

- From among these combinations we select just those which have a required RL parameter – another reduction of the number of combinations.
- The consequence: the combinations which contain four „0“ and six „1“ (and vice versa) must be used as well (not just five „0“ and five „1“).
- **If we use such combinations then the requirement of zero DC component is not satisfied – it must be solved.**

- The solution:
  - In the set of 10 bit characters, there are 210 charecters which contain six „0“ and four „1“.
  - These characters have also their inverse form, i. e. six „1“ and four „0“.
  - It is true that these characters have not zero DC component (because they do not contain the same number of „0“ and „1“).

- Therefore, the number of „0“ and „1“ must be checked and if it is not equal, then a compensation must be done during the communication.
- Disparity concept – the difference between transmitted „0“ and „1“.
- Negative disparity – the number of transmitted „0“ is higher than the number of transmitted „1“.
- Current disparity – immediate disparity.
- Note.: a character with five „0“ and five „1“ does not change the current disparity.
- A character with six „1“ and four „0“ (and vice versa) changes the current disparity.

- The principle:
  - Every combination has two values assigned, if the current disparity is negative, then one of 10 bit values is used, if it is positive then the other value is used.

- Example:
  - The current disparity is not balanced, then the code for the negative current disparity contains six „1“ and four „0“, for the positive current disparity the opposite value. Based on the value of the current disparity, one of these values is used.

- Example from the table:
  - The combination 00h has two 10-bit values assigned: 1001110100 a 0110001011, both values have the same number of „0“ and „1“, so that they do not change the current disparity.
  - The combination 06h has also two 10-bit values assigned: 0110011011 or 0110010100, both values have different number of „0“ a „1“, so that they modify the current disparity.

- By choosing proper value from the table it is guaranteed that the encoded sequence of bits is balanced and has also the required value of RL.

- The backward procedure (accepting data by the receiver): first of all the accepted information must be separated into 10-bit segments and then backward decoding must be done.

Encoding table

		Output	
Name	**Byte**	**Current rd–**	**Current rd+**
D0.0	00h	100111 0100	011000 1011
D1.0	01h	011101 0100	100010 1011
D2.0	02h	101101 0100	010010 1011
D3.0	03h	110001 1011	110001 0100
D4.0	04h	110101 0100	001010 1011
D5.0	05h	101001 1011	101001 0100
D6.0	06h	011001 1011	011001 0100
D7.0	07h	111000 1011	000111 0100
D8.0	08h	111001 0100	000110 1011
D9.0	09h	100101 1011	100101 0100
D10.0	0Ah	010101 1011	010101 0100
D11.0	0Bh	110100 1011	110100 0100
D12.0	0Ch	001101 1011	001101 0100
D13.0	0Dh	101100 1011	101100 0100
D14.0	0Eh	011100 1011	011100 0100
D15.0	0Fh	010111 0100	101000 1011

- The concept: rd – running disparity

- An example: sending HELLO message

ASCII character	Code	Starting disparity	10 bit value	Final disparity
H	48h	-	1110010101	+
E	45h	+	1010010101	+
L	4Ch	+	0011010101	+
L	4Ch	+	0011010101	+
O	4Fh	+	1010000101	-

Concepts: starting disparity, final disparity

- Principles:
  - The byte with six „1“ and four „0“ will set positive disparity.
  - The byte with six „0“ and four „1“ will set negative disparity.
  - The byte with five „1“ and five „0“ does not change the disparity.

- The encoded message has the following form:
- 1110010101 1010010101 0011010101 0011010101 1010000101
- **The result: twenty five „0“ and twenty five „1“.**

- Summary - 8b/10b encoding:
  - From among 1024 combinations, such combinations are selected which satisfy both requirements, i. e. required RL and zero DC component.
  - We have 252 such combinations (they contain five „0“ and five „1“).
  - The remaining combinations are chosen with the requirement to offer two alternatives of „0“ and „1“ combinations (6 or 4 and vice versa).
- How data are checked by the receiver:
  - The running disparity is checked. Let us see the example below – the assumption: there is a mistake in accepted data which arose

by changing one bit value into opposite value. The consequence: instead of „H“, „E“ was recognised.

ASCII znak	8 bit code	running disparity	Transmitted 10 bits	Received 10 bits	Running disparity	Decoded 8 bit value	ASCII character
		-			-		
H	48h		1110010101	1010010101			E
		+			-		
E	45h		1010010101	1010010101			E
		+			-		
L	4Ch		0011010101	0011010101			L
		+			-		
L	4Ch		0011010101	0011010101			L
		+			-		
O	4Fh		1010000101	1010000101		ERROR	
		-			-		

- The conclusion:
  - An error in one bit will cause another error (incorrect running disparity), the character „O“ should have according to the value of running disparity the opposite value of its 10 bit value (i.e. six „1“ and four „0“).
  - A character is accepted which supports the negative disparity, it is seen as an error.
  - Altogether twenty four „1“ and twenty six „0“ were accepted (the last accepted character supported negative disparity instead of changing it to positive disparity).
  - **The conclusion: according to the numer of „0“ a „1“ in the accepted message, an error is identified**.